

WHAT IS CLAIMED IS:

1 1. A concurrent shared object representation comprising:
2 a computer readable encoding for a sequence of zero or more values; and
3 access operations defined for access to each of opposing ends of the sequence,
4 wherein execution of any one of the access operations is non-blocking with
5 respect to any other execution of the access operations throughout a
6 complete range of valid states, including one or more boundary
7 condition states, and
8 wherein, at least for those of the valid states other than the one or more
9 boundary condition states, opposing-end ones of the access operations
10 are disjoint.

1 2. The concurrent shared object representation of claim 1,
2 wherein the computer readable encoding includes an array of elements for
3 representing the sequence; and
4 wherein the one or more boundary condition states include a full state and an
5 empty state.

1 3. The concurrent shared object representation of claim 1,
2 wherein the computer readable encoding includes a linked-list of nodes
3 representing the sequence; and
4 wherein the one or more boundary condition states include one or more empty
5 states.

1 4. The concurrent shared object representation of claim 1, wherein the access
2 operations include push and pop operations.

1 5. The concurrent shared object representation of claim 4, wherein the access
2 operations further include delete operations.

1 6. The concurrent shared object representation of claim 1, wherein the access
2 operations include push and pop operations, including opposing end variants of each.

1 7. The concurrent shared object representation of claim 1, wherein the access
2 operations include push and pop operations, including opposing end variants of at
3 least one of the push and pop operations.

4 8. The concurrent shared object representation of claim 2,
5 wherein the array of elements is organized as a circular buffer of fixed size
6 with opposing-end indices respectively identifying opposing ends of
7 the sequence; and
8 wherein concurrent non-blocking access is mediated, at least in part, by
9 performing, during execution of each of the access operations, an
10 atomic update of a respective one of the opposing-end indices and of
11 an array element corresponding thereto.

1 9. The concurrent shared object representation of claim 3,
2 wherein the access operations include push, pop and delete operations, and
3 wherein concurrent access is mediated, at least in part, by performing, during
4 execution of each of the pop operations, an atomic update of a list node
5 and both a deleted node indication and list-end identifier corresponding
6 thereto.

1 10. The concurrent shared object representation of claim 9,
2 wherein concurrent access is further mediated, at least in part, by performing,
3 during execution of each of the delete operations, an atomic update of
4 a deleted node indication and at least one list-end identifier
5 corresponding thereto.

1 11. The concurrent shared object representation of claim 3, wherein the
2 linked-list of nodes is a doubly-linked list thereof.

1 12. A method of managing access to a dynamically allocated list susceptible
2 to concurrent operations on a sequence encoded therein, the method comprising:
3 executing as part of a pop operation, an atomic update of a list node and both a
4 deleted node indication and list-end identifier corresponding thereto;

5 the deleted node indication marking the corresponding element for subsequent
6 deletion from the list.

1 13. The method of claim 12, further comprising:
2 executing as part of a delete operation, an atomic update of a deleted node
3 indication and at least one list-end identifier corresponding thereto.

1 14. The method of claim 12, further comprising:
2 responsive to the deleted node indication, excising a marked node from the list
3 by atomically updating opposing direction pointers impinging thereon
4 and the deleted node indication thereto.

1 15. The method of claim 12, further comprising:
2 deleting the marked element from the list at least before completion of a same-
3 end push or pop operation.

1 16. The method of claim 13,
2 wherein the list is a doubly-linked list susceptible to concurrent operation of
3 opposing-end variants of the pop operation; and
4 wherein the atomic update includes execution of a DCAS.

1 17. The method of claim 13,
2 wherein the list is a doubly-linked list susceptible to concurrent operation of a
3 same-end push operation; and
4 wherein the atomic update includes execution of a DCAS.

1 18. The method of claim 12, further comprising:
2 wherein the deleted node indication is encoded integral with an end-node
3 identifying pointer.

1 19. The method of claim 12, further comprising:
2 wherein the deleted node indication is encoded as a dummy node.

1 20. A computer program product encoded in at least one computer readable
2 medium, the computer program product comprising:
3 at least one functional sequence providing non-blocking access to on a
4 concurrent shared object, the concurrent shared object instantiable as a
5 linked-list delimited by a pair of end identifiers;
6 wherein instances of the at least one functional sequence concurrently
7 executable by plural processors of a multiprocessor and each include
8 an atomic operation to atomically update one of the end identifiers and
9 a node of the linked-list corresponding thereto,
10 wherein for opposing end instances, the atomic updates are disjoint for at least
11 all non-empty states of the concurrent shared object.

1 21. A computer program product as recited in 20, wherein the at least one
2 functional sequence includes both push and pop functional sequences.

1 22. A computer program product as recited in 20,
2 wherein the at least one computer readable medium is selected from the set of
3 a disk, tape or other magnetic, optical, or electronic storage medium
4 and a network, wireline, wireless or other communications medium.

1 23. An apparatus comprising:
2 plural processors;
3 a store addressable by each of the plural processors;
4 first- and second-end identifier stores accessible to each of the plural
5 processors for identifying opposing ends of a concurrent shared object
6 in the addressable store; and
7 means for coordinating competing pop operations, the coordinating means
8 employing in each instance thereof, an atomic operation to
9 disambiguate a retry state and a boundary condition state of the
10 concurrent shared object based on then-current contents of one, but not
11 both, of the first- and second-end identifier stores and an element of
12 the concurrent shared object corresponding thereto.